

EFFICIENT COMPUTATION OF TERMS OF LINEAR RECURRENCE SEQUENCES OF ANY ORDER

Dmitry I. Khomovsky

Lomonosov Moscow State University, Moscow, RF

khomovskij@physics.msu.ru

Received: , Revised: , Accepted: , Published:

Abstract

In this paper we give efficient algorithms for computing second-, third-, and fourth-order linear recurrences. We also present an algorithm scheme for computing terms with the indices $N, \dots, N+n-1$ of an n th-order linear recurrence. Unlike Fiduccia's algorithm [1] our approach uses certain formulas for modular polynomial squarings.

1. Introduction

The Lucas sequences $\{U_k(P, Q)\}$ and $\{V_k(P, Q)\}$ are defined recursively by

$$f_{k+2} = Pf_{k+1} - Qf_k, \quad (1)$$

with the initial values¹ $U_0 = 0, U_1 = 1, V_0 = 2, V_1 = P$. The characteristic equation of the recurrence relation (1) is $x^2 - Px + Q = 0$. Its roots are $\alpha = \frac{P+\sqrt{\Delta}}{2}$ and $\bar{\alpha} = \frac{P-\sqrt{\Delta}}{2}$, where $\Delta = P^2 - 4Q$. The Lucas sequences can be expressed in terms of α and $\bar{\alpha}$ as follows:

$$U_k = \frac{\alpha^k - \bar{\alpha}^k}{\alpha - \bar{\alpha}}, \quad V_k = \alpha^k + \bar{\alpha}^k, \quad P = \alpha + \bar{\alpha}, \quad Q = \alpha\bar{\alpha}, \quad \sqrt{\Delta} = \alpha - \bar{\alpha}. \quad (2)$$

Let $\{W_k(a_0, \dots, a_{n-1}; p_0 \dots p_{n-1})\}$ be an n th-order linear recurrence defined by the relation

$$f_{k+n} = p_0 f_{k+n-1} + p_1 f_{k+n-2} + \dots + p_{n-1} f_k, \quad (3)$$

with initial values $W_i = a_i$ ($0 \leq i \leq n-1$). The characteristic polynomial is:

$$g(x) = x^n - (p_0 x^{n-1} + p_1 x^{n-2} + \dots + p_{n-1}). \quad (4)$$

Computation of linear recurrences has been studied by many authors [1, 2, 3]. The most effective algorithm was proposed by Fiduccia in 1985. To compute the

¹In this paper instead of $U_k(P, Q)$ and $V_k(P, Q)$ we will write U_k and V_k if it is not ambiguous.

N th term of an n th-order linear recurrence with the help of this method first we need to compute $r(x) = x^N \bmod g(x)$, where $g(x) = x^n - \sum_{i=0}^{n-1} p_i x^{n-1-i}$ is the characteristic polynomial. Then we need to compute $r(C)$, where C is the $n \times n$ companion matrix of the linear recurrence:

$$C = \begin{pmatrix} 0 & & & p_{n-1} \\ 1 & & & p_{n-2} \\ & 1 & & p_{n-3} \\ & & \ddots & \vdots \\ & & & 1 & p_0 \end{pmatrix}. \quad (5)$$

Finally, we multiply the row vector of initial values (a_0, \dots, a_{n-1}) by the first column of $r(C)$ and obtain the N th term. The computational complexity of this algorithm is $O(\mu(n) \log N)$. Here, $\mu(n)$ is the number of operations required to multiply two polynomials of degree $n - 1$.

2. Computation of second-order linear recurrences

Let the second-order linear recurrence sequence $\{W_k\}$ be defined by the relation² $W_{k+2} = PW_{k+1} - QW_k$, with $W_0 = A, W_1 = B$. It was intensively studied by Horadam [8, 9].

The matrix method is often used to prove some identities concerning the generalized Fibonacci and Lucas numbers [5, 6]. For the Lucas sequences we have the following matrix formula:

$$\begin{pmatrix} U_{k+1} & V_{k+1} \\ U_k & V_k \end{pmatrix} = M \begin{pmatrix} U_k & V_k \\ U_{k-1} & V_{k-1} \end{pmatrix}, \quad \text{where } M = \begin{pmatrix} P & -Q \\ 1 & 0 \end{pmatrix}. \quad (6)$$

Then

$$\begin{pmatrix} U_{k+1} & V_{k+1} \\ U_k & V_k \end{pmatrix} = M^k \begin{pmatrix} 1 & P \\ 0 & 2 \end{pmatrix}. \quad (7)$$

Lemma 1. *Let the sequence $\{W_k(A, B; P, Q)\}$ be defined by the relation $W_{k+2} = PW_{k+1} - QW_k$, with initial values $W_0 = A, W_1 = B$. Then*

$$W_k = BU_k - AQU_{k-1}, \quad (8)$$

$$W_k = (B - AP)U_k + AU_{k+1}. \quad (9)$$

Proof. We have:

$$\begin{aligned} \begin{pmatrix} W_{k+1} \\ W_k \end{pmatrix} &= M^k \begin{pmatrix} B \\ A \end{pmatrix} = BM^k \begin{pmatrix} 1 \\ 0 \end{pmatrix} + AM^k \begin{pmatrix} 0 \\ 1 \end{pmatrix} = B \begin{pmatrix} U_{k+1} \\ U_k \end{pmatrix} + AM^{k-1} \begin{pmatrix} -Q \\ 0 \end{pmatrix} \\ &= \begin{pmatrix} BU_{k+1} - AQU_k \\ BU_k - AQU_{k-1} \end{pmatrix}. \end{aligned}$$

²For recurrences of order greater than 2 we will use the relation (3).

From this we get (8). By the definition of the Lucas sequence $QU_{k-1} = PU_k - U_{k+1}$. Using this we obtain (9). \square

Computation of the second-order linear recurrence $\{W_k(A, B; P, Q)\}$ can be done by the Lucas sequence $\{U_k(P, Q)\}$. In a sense, the sequence $\{U_k\}$ is a basic. This result is well-known, see [8]. By Lemma 1 we get the following classical identity that will be required in the future:

$$V_k = PU_k - 2QU_{k-1}. \quad (10)$$

We will use the notation:

$$\begin{aligned} S &= \begin{pmatrix} 1 & P \\ 0 & 2 \end{pmatrix}, [MS]_k = \begin{pmatrix} U_{k+1} & V_{k+1} \\ U_k & V_k \end{pmatrix}, [MSU]_k = \begin{pmatrix} U_{k+1} \\ U_k \end{pmatrix}, \\ S^{-1} &= \begin{pmatrix} 1 & -P/2 \\ 0 & 1/2 \end{pmatrix}, M^{-1} = \begin{pmatrix} 0 & 1 \\ -1/Q & P/Q \end{pmatrix}. \end{aligned} \quad (11)$$

Here, S denotes the initial values matrix.

Theorem 1. *Let $\{U_k(P, Q)\}$ be the Lucas sequence. Then*

$$\begin{pmatrix} U_{mk+1} \\ U_{mk} \end{pmatrix} = \begin{pmatrix} U_{k+1} & -QU_k \\ U_k & U_{k+1} - PU_k \end{pmatrix}^{m-1} \begin{pmatrix} U_{k+1} \\ U_k \end{pmatrix}. \quad (12)$$

Proof. We have

$$\begin{aligned} \begin{pmatrix} U_{mk+1} \\ U_{mk} \end{pmatrix} &= M^{mk} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \underbrace{M^k S S^{-1} M^k S S^{-1} \dots M^k S S^{-1}}_{m-1 \text{ times}} M^k \begin{pmatrix} 1 \\ 0 \end{pmatrix} \\ &= (M^k S S^{-1})^{m-1} \begin{pmatrix} U_{k+1} \\ U_k \end{pmatrix}. \end{aligned} \quad (13)$$

By (11) we get

$$[MSU]_{mk} = ([MS]_k S^{-1})^{m-1} [MSU]_k. \quad (14)$$

Now we calculate $[MS]_k S^{-1}$:

$$[MS]_k S^{-1} = \begin{pmatrix} U_{k+1} & (-PU_{k+1} + V_{k+1})/2 \\ U_k & (-PU_k + V_k)/2 \end{pmatrix}. \quad (15)$$

With the help of (10) we eliminate V_k, V_{k+1} :

$$[MS]_k S^{-1} = \begin{pmatrix} U_{k+1} & -QU_k \\ U_k & -QU_{k-1} \end{pmatrix}. \quad (16)$$

Since $-QU_{k-1} = U_{k+1} - PU_k$ by the definition of the Lucas sequences, we have

$$[MS]_k S^{-1} = \begin{pmatrix} U_{k+1} & -QU_k \\ U_k & U_{k+1} - PU_k \end{pmatrix}. \quad (17)$$

We note that the elements of $[MS]_k S^{-1}$ are linearly related to the terms U_k, U_{k+1} . Finally, we can modify (13) into (12). \square

If $m = 2$ in (12), then we obtain the following identities:

$$U_{2k} = U_k(2U_{k+1} - PU_k), \quad (18)$$

$$U_{2k+1} = U_{k+1}^2 - QU_k^2. \quad (19)$$

If we replace k by $k + 1$ in (18) and use $U_{k+2} = PU_{k+1} - QU_k$, then we obtain

$$U_{2k+2} = U_{k+1}(PU_{k+1} - 2QU_k). \quad (20)$$

Now using (18), (19), and (20) we can present an algorithm for computing two terms of $\{U_k(P, Q)\}$ with the indices N and $N + 1$. We need four temporary memories: u_1, u_2, U_1, U_2 .

Algorithm 1 Computing the Lucas sequence $\{U_k(P, Q)\}$

Input: $N = \sum_{i=0}^{m-1} b_i 2^i$, $(b_{m-1} = 1)$
 P, Q

Output: U_N, U_{N+1}

```

1:  $U_1 \leftarrow 1; U_2 \leftarrow P$ 
2: for  $j$  from  $m - 2$  to  $0$  by  $-1$  do
3:    $u_1 \leftarrow U_1; u_2 \leftarrow U_2$ 
4:   if  $b_j = 1$  then
5:      $U_1 \leftarrow u_2^2 - Qu_1^2; U_2 \leftarrow u_2(Pu_2 - 2Qu_1)$ 
6:   else if
7:      $U_1 \leftarrow u_1(2u_2 - Pu_1); U_2 \leftarrow u_2^2 - Qu_1^2$ 
8:   end if
9: end for
11: return  $U_1, U_2$ 

```

Remark. Such a computing method was discussed in [5]. Probably it did not become widely known because [5] does not contain the algorithm.

Suppose we have computed U_N, U_{N+1} by Algorithm 1, then with the help of (9) we get W_N . Using $W_{N+1} = BU_{N+1} - AQU_N$ we get W_{N+1} . Thus, in a general case to compute the terms U_N, U_{N+1} and W_N, W_{N+1} we need $3m$ multiplications³, here $m = \lfloor \log_2 N \rfloor + 1$. But when $Q = 1$ or $Q = a^2$, we can slightly transform Algorithm 1 so that we need only $2m$ multiplications. Indeed, when $Q = 1$, we can to replace the expression $u_2^2 - u_1^2$ by $(u_2 - u_1)(u_2 + u_1)$ at steps 5, 7. When $Q = a^2$, we use the formula $u_2^2 - Qu_1^2 = (u_2 - au_1)(u_2 + au_1)$.

Algorithm 1 is good for computing Lucas sequences over \mathbb{Z}_n . Moreover, we need the same number of multiplications as in computation over \mathbb{Z} .

³We imply that P, Q are not large. So multiplications that involve them are similar to additions.

2.1. Comparison with other existing algorithms

Currently, the main algorithm [7] for quick computation of the Lucas sequence terms U_N , V_N uses the following properties:

$$V_{2k+1} = V_{k+1}V_k - PQ^k, \quad V_{2k} = V_k^2 - 2Q^k \quad (21)$$

$$U_{2k+1} = U_{k+1}V_k - Q^k, \quad U_{2k} = U_kV_k. \quad (22)$$

When $Q = \pm 1$, the algorithm needs $3m$ multiplications. When $Q \neq \pm 1$ and without any assumptions about N , this algorithm needs $11m/2$ multiplications. We see that Algorithm 1 is more effective for computing an arbitrary second-order linear recurrence, but there is an important case when the algorithm offered in [7] is better. This is so when we need to compute only $V_N(P, 1)$ or $V_N(P, -1)$. For $N = 2^s(2d+1)$ the algorithm in [7] needs $2\lceil \log_2(2d+1) \rceil + s$ multiplications while Algorithm 1 needs $2\lceil \log_2(2d+1) \rceil + 2s$. So in applications such as Lucas-based cryptosystem [11] and Lucas-Lehmer-Riesel primality test [10] it is preferable to use the algorithm offered in [7].

Now we compare Algorithm 1 with Fiduccia's algorithm. The characteristic polynomial is $g(x) = x^2 - Px + Q$. To compute $x^N \bmod g(x)$ Fiduccia's algorithm uses classical method of repeating squaring. For an arbitrary linear polynomial $h(x) = -u_1x + u_2$ we have $h^2(x) \bmod g(x) = -u_1(2u_2 - Pu_1)x + u_2^2 - Qu_1^2$. As is seen from this result we can use the formulas (18), (19) for modular polynomial squarings. Therefore, Algorithm 1 together with the formula (9) is one way of implementing Fiduccia's algorithm for second-order linear recurrences, where is used the explicit formulas for modular polynomial squarings.

3. Computation of third-order linear recurrences

We will follow the notation for third-order linear recurrences according to [12]. The sequences $\{X_k(p, q, r)\}$, $\{Y_k(p, q, r)\}$, and $\{Z_k(p, q, r)\}$ are defined recursively by

$$f_{k+3} = pf_{k+2} + qf_{k+1} + rf_k, \quad (23)$$

with the initial values $X_0 = 0$, $X_1 = 0$, $X_2 = 1$, $Y_0 = 0$, $Y_1 = 1$, $Y_2 = 0$, $Z_0 = 1$, $Z_1 = 0$, $Z_2 = 0$. Similar to (7) we have

$$\begin{pmatrix} X_{k+2} & Y_{k+2} & Z_{k+2} \\ X_{k+1} & Y_{k+1} & Z_{k+1} \\ X_k & Y_k & Z_k \end{pmatrix} = M^k S, \text{ where } M = \begin{pmatrix} p & q & r \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}, \quad S = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}. \quad (24)$$

Lemma 2. *Let the sequence $\{W_k(a_0, a_1, a_2; p, q, r)\}$ be defined by the relation*

$$W_{k+3} = pW_{k+2} + qW_{k+1} + rW_k, \quad (25)$$

with the initial values $W_0 = a_0, W_1 = a_1, W_2 = a_2$. Then

$$W_k = a_2 X_k + (a_1 q + a_0 r) X_{k-1} + a_1 r X_{k-2}, \quad (26)$$

$$W_k = (a_2 - a_1 p - a_0 q) X_k + (a_1 - a_0 p) X_{k+1} + a_0 X_{k+2}. \quad (27)$$

Proof. We have:

$$\begin{aligned} \begin{pmatrix} W_{k+2} \\ W_{k+1} \\ W_k \end{pmatrix} &= M^k \begin{pmatrix} a_2 \\ a_1 \\ a_0 \end{pmatrix} = a_2 M^k \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} + a_1 M^k \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} + a_0 M^k \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \\ &= a_2 \begin{pmatrix} X_{k+2} \\ X_{k+1} \\ X_k \end{pmatrix} + a_1 M^{k-1} \begin{pmatrix} q \\ 0 \\ 1 \end{pmatrix} + a_0 M^{k-1} \begin{pmatrix} r \\ 0 \\ 0 \end{pmatrix} \\ &= a_2 \begin{pmatrix} X_{k+2} \\ X_{k+1} \\ X_k \end{pmatrix} + (a_1 q + a_0 r) \begin{pmatrix} X_{k+1} \\ X_k \\ X_{k-1} \end{pmatrix} + a_1 M^{k-2} \begin{pmatrix} r \\ 0 \\ 0 \end{pmatrix} \\ &= \begin{pmatrix} a_2 X_{k+2} + (a_1 q + a_0 r) X_{k+1} + a_1 r X_k \\ a_2 X_{k+1} + (a_1 q + a_0 r) X_k + a_1 r X_{k-1} \\ a_2 X_k + (a_1 q + a_0 r) X_{k-1} + a_1 r X_{k-2} \end{pmatrix}. \end{aligned} \quad (28)$$

So we obtain (26). With the help of $X_{k-2} = (X_{k+1} - pX_k - qX_{k-1})/r$ and $X_{k-1} = (X_{k+2} - pX_{k+1} - qX_k)/r$ we get (27). \square

By Lemma 2 we get the following identities, some of which can be found in [12],

$$Y_k = qX_{k-1} + rX_{k-2}, \quad (29)$$

$$Z_k = rX_{k-1}, \quad (30)$$

$$Y_k = X_{k+1} - pX_k, \quad (31)$$

$$Z_k = X_{k+2} - pX_{k+1} - qX_k. \quad (32)$$

Further we will use the notation:

$$[MS]_k = \begin{pmatrix} X_{k+2} & Y_{k+2} & Z_{k+2} \\ X_{k+1} & Y_{k+1} & Z_{k+1} \\ X_k & Y_k & Z_k \end{pmatrix}, \quad [MSX]_k = \begin{pmatrix} X_{k+2} \\ X_{k+1} \\ X_k \end{pmatrix}. \quad (33)$$

Now we give the theorem that is an analogous to Theorem 1.

Theorem 2. Let $\{X_k(p, q, r)\}$ be a third-order linear recurrence sequence with the initial values $X_0 = 0, X_1 = 0, X_2 = 1$. Then

$$\begin{pmatrix} X_{mk+2} \\ X_{mk+1} \\ X_{mk} \end{pmatrix} = \begin{pmatrix} X_{k+2} & qX_{k+1} + rX_k & rX_{k+1} \\ X_{k+1} & X_{k+2} - pX_{k+1} & rX_k \\ X_k & X_{k+1} - pX_k & X_{k+2} - pX_{k+1} - qX_k \end{pmatrix}^{m-1} \begin{pmatrix} X_{k+2} \\ X_{k+1} \\ X_k \end{pmatrix}. \quad (34)$$

Proof. We have

$$\begin{aligned} \begin{pmatrix} X_{mk+2} \\ X_{mk+1} \\ X_{mk} \end{pmatrix} &= M^{mk} \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} = \underbrace{M^k SS^{-1} M^k SS^{-1} \dots M^k SS^{-1} M^k}_{m-1 \text{ times.}} \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \\ &= (M^k SS^{-1})^{m-1} \begin{pmatrix} X_{k+2} \\ X_{k+1} \\ X_k \end{pmatrix} = [MS]_k^{m-1} [MSX]_k. \end{aligned} \quad (35)$$

With the help of (29) – (32) we eliminate the terms $Y_k, Y_{k+1}, Y_{k+2}, Z_k, Z_{k+1}, Z_{k+2}$ from $[MS]_k$. It may be done in such a way that $[MS]_k$ contains only X_k, X_{k+1}, X_{k+2} .

$$[MS]_k = \begin{pmatrix} X_{k+2} & qX_{k+1} + rX_k & rX_{k+1} \\ X_{k+1} & X_{k+2} - pX_{k+1} & rX_k \\ X_k & X_{k+1} - pX_k & X_{k+2} - pX_{k+1} - qX_k \end{pmatrix}. \quad (36)$$

Finally, we can modify (35) into (34). \square

If we put $m = 2$ in (34), then we get the following formulas:

$$X_{2k+2} = X_{k+2}^2 + X_{k+1}(qX_{k+1} + 2rX_k), \quad (37)$$

$$X_{2k+1} = rX_k^2 + X_{k+1}(2X_{k+2} - pX_{k+1}), \quad (38)$$

$$X_{2k} = X_{k+1}^2 + X_k(2X_{k+2} - 2pX_{k+1} - qX_k). \quad (39)$$

Remark. If we calculate the remainder

$$(c_1x^2 + c_2x + c_3)^2 \bmod x^3 - px^2 - qx - r, \quad (40)$$

then we obtain the formulas similar (but not the same) to (37) – (39) for squaring of quadratic polynomials modulo $g(x) = x^3 - px^2 - qx - r$. They can be used in Fiduccia's algorithm for computing third-order recurrences.

Remark. If we put $r = 0, q = -Q$ in these formulas and subtract 1 from all indices, then up to the substitution of U for X we obtain the identities for second-order recurrences. It follows from $X_{k+1}(P, -Q, 0) = U_k(P, Q)$.

To get an algorithm for computing X_N, X_{N+1}, X_{N+2} similar to the binary exponentiation we need to be able to compute $X_{2k}, X_{2k+1}, X_{2k+2}, X_{2k+3}$ using X_k, X_{k+1}, X_{k+2} . So we need another formula that helps us to compute X_{2k+3} . It can be obtained from (38) if we replace k by $k+1$ and use $X_{k+3} = pX_{k+2} + qX_{k+1} + rX_k$. It is as follows:

$$X_{2k+3} = rX_{k+1}^2 + X_{k+2}(pX_{k+2} + 2qX_{k+1} + 2rX_k). \quad (41)$$

Now we present an algorithm for computing third-order recurrences based on the formulas (37) – (41). We need to use six temporary memories.

Algorithm 2 Computing the third-order linear recurrence $\{X_k(p, q, r)\}$

Input: $N = \sum_{i=0}^{m-1} b_i 2^i$, ($b_{m-1} = 1$)
 p, q, r

Output: X_N, X_{N+1}, X_{N+2}

```

1:  $X_1 \leftarrow 0; X_2 \leftarrow 1; X_3 \leftarrow p$ 
2: for  $j$  from  $m-2$  to  $0$  by  $-1$  do
3:    $x_1 \leftarrow X_1; x_2 \leftarrow X_2; x_3 \leftarrow X_3$ 
4:   if  $b_j = 1$  then
5:      $X_1 \leftarrow rx_1^2 + x_2(2x_3 - px_2); X_2 \leftarrow x_3^2 + x_2(qx_2 + 2rx_1);$ 
        $X_3 \leftarrow rx_2^2 + x_3(px_3 + 2qx_2 + 2rx_1)$ 
6:   else if
7:      $X_1 \leftarrow x_2^2 + x_1(2x_3 - 2px_2 - qx_1); X_2 \leftarrow rx_1^2 + x_2(2x_3 - px_2);$ 
        $X_3 \leftarrow x_3^2 + x_2(qx_2 + 2rx_1)$ 
8:   end if
9: end for
11: return  $X_1, X_2, X_3$ 

```

Algorithm 1 needs $3m$ multiplications and $3m$ squarings. In [13] it was shown that the exponentiation $y = \alpha^N$ in the cubic extension field $GF(q^3)$ can be computed with about $4m$ multiplications and $4m$ squarings in $GF(q)$. Using Algorithm 1 this can be done more effectively.

4. Computation of fourth-order linear recurrences

Since this section is similar to Section 4, we give only the main formulas and the final algorithm.

The fourth-order linear recurrence $\{W_k(a_0, a_1, a_2, a_3; p_0, p_1, p_2, p_3)\}$ is defined recursively by

$$f_{k+4} = p_0 f_{k+3} + p_1 f_{k+2} + p_2 f_{k+1} + p_3 f_k, \quad (42)$$

with the initial values $W_0 = a_0, W_1 = a_1, W_2 = a_2, W_3 = a_3$. Denote the sequence $\{W_k(0, 0, 0, 1; p_0, p_1, p_2, p_3)\}$ by $\{X_k(p_0, p_1, p_2, p_3)\}$. The formulas which can be obtained by the matrix method are:

$$W_k = a_3 X_k + (a_0 p_3 + a_1 p_2 + a_2 p_1) X_{k-1} + (a_1 p_3 + a_2 p_2) X_{k-2} + a_2 p_3 X_{k-3}, \quad (43)$$

$$W_k = (a_3 - a_2 p_0 - a_1 p_1 - a_0 p_2) X_k + (a_2 - a_1 p_0 - a_0 p_1) X_{k+1} + (a_1 - a_0 p_0) X_{k+2} + a_0 X_{k+3}. \quad (44)$$

Therefore, the sequence $\{X_k\}$ may be chosen as a basis for fourth-order linear recurrences.

We will use $W_k(a_0, a_1, a_2, a_3)$ instead of $W_k(a_0, a_1, a_2, a_3; p_0, p_1, p_2, p_3)$. By (43), (44), and (42) we obtain the following

$$W_k(0, 0, 1, 0) = p_1 X_{k-1} + p_2 X_{k-2} + p_3 X_{k-3}, \quad (45)$$

$$W_k(0, 1, 0, 0) = p_2 X_{k-1} + p_3 X_{k-2}, \quad (46)$$

$$W_k(1, 0, 0, 0) = p_3 X_{k-1}, \quad (47)$$

$$W_k(0, 0, 1, 0) = -p_0 X_k + X_{k+1}, \quad (48)$$

$$W_k(0, 1, 0, 0) = -p_1 X_k - p_0 X_{k+1} + X_{k+2}, \quad (49)$$

$$W_k(1, 0, 0, 0) = -p_2 X_k - p_1 X_{k+1} - p_0 X_{k+2} + X_{k+3}. \quad (50)$$

For convenience, we use the notation ${}^i W_k$ for $W_k(a_0, a_1, a_2, a_3)$ with only one nonzero $a_i = 1$. Then by the matrix method we get:

$$\begin{pmatrix} X_{mk+3} \\ X_{mk+2} \\ X_{mk+1} \\ X_{mk} \end{pmatrix} = \begin{pmatrix} X_{k+3} & {}^2 W_{k+3} & {}^1 W_{k+3} & {}^0 W_{k+3} \\ X_{k+2} & {}^2 W_{k+2} & {}^1 W_{k+2} & {}^0 W_{k+2} \\ X_{k+1} & {}^2 W_{k+1} & {}^1 W_{k+1} & {}^0 W_{k+1} \\ X_k & {}^2 W_k & {}^1 W_k & {}^0 W_k \end{pmatrix}^{m-1} \begin{pmatrix} X_{k+3} \\ X_{k+2} \\ X_{k+1} \\ X_k \end{pmatrix}. \quad (51)$$

With the help of (45) – (50) we transform this matrix and have

$$\begin{pmatrix} X_{k+3} & p_1 X_{k+2} + p_2 X_{k+1} + p_3 X_k & p_2 X_{k+2} + p_3 X_{k+1} & p_3 X_{k+2} \\ X_{k+2} & X_{k+3} - p_0 X_{k+2} & p_2 X_{k+1} + p_3 X_k & p_3 X_{k+1} \\ X_{k+1} & X_{k+2} - p_0 X_{k+1} & X_{k+3} - p_0 X_{k+2} - p_1 X_{k+1} & p_3 X_k \\ X_k & X_{k+1} - p_0 X_k & X_{k+2} - p_0 X_{k+1} - p_1 X_k & R_{4,4} \end{pmatrix}. \quad (52)$$

Here, $R_{4,4} = X_{k+3} - p_0 X_{k+2} - p_1 X_{k+1} - p_2 X_k$. As is seen this matrix has a special structure. This will be used in the next section.

Finally, if we put $m = 2$ and simplify (51) we get the following formulas:

$$X_{2k+3} = X_{k+3}^2 + X_{k+2}(p_1 X_{k+2} + 2p_2 X_{k+1} + 2p_3 X_k) + p_3 X_{k+1}^2, \quad (53)$$

$$X_{2k+2} = X_{k+2}(2X_{k+3} - p_0 X_{k+2}) + X_{k+1}(p_2 X_{k+1} + 2p_3 X_k), \quad (54)$$

$$X_{2k+1} = X_{k+2}^2 + X_{k+1}(2X_{k+3} - 2p_0 X_{k+2} - p_1 X_{k+1}) + p_3 X_k^2, \quad (55)$$

$$X_{2k} = X_{k+1}(2X_{k+2} - p_0 X_{k+1}) + X_k(2X_{k+3} - 2p_0 X_{k+2} - 2p_1 X_{k+1} - p_2 X_k). \quad (56)$$

We also need the formula for X_{2k+4} . It can be obtained from (54) if we replace k by $k+1$ and use (42) for X_{k+4} . It is as follows:

$$X_{2k+4} = X_{k+3}(p_0 X_{k+3} + 2p_1 X_{k+2} + 2p_2 X_{k+1} + 2p_3 X_k) + X_{k+2}(p_2 X_{k+2} + 2p_3 X_{k+1}). \quad (57)$$

Remark. If we put $p_3 = 0$, $p_2 = r$, $p_1 = q$, $p_0 = p$ in (53)-(57) and subtract 1 from all indices, then we obtain the identities for third-order recurrences.

Algorithm 3 Computing the fourth-order linear recurrence $\{X_k(p_0, p_1, p_2, p_3)\}$

Input: $N = \sum_{i=0}^{m-1} b_i 2^i$, ($b_{m-1} = 1$)

p_0, p_1, p_2, p_3

Output: $X_N, X_{N+1}, X_{N+2}, X_{N+3}$

```

1:  $X_1 \leftarrow 0; X_2 \leftarrow 0; X_3 \leftarrow 1; X_4 \leftarrow p_0$ 
2: for  $j$  from  $m-2$  to  $0$  by  $-1$  do
3:  $x_1 \leftarrow X_1; x_2 \leftarrow X_2; x_3 \leftarrow X_3; x_4 \leftarrow X_4$ 
4:   if  $b_j = 1$  then
5:      $X_1 \leftarrow x_3^2 + x_2(2x_4 - 2p_0x_3 - p_1x_2) + p_3x_1^2;$ 
        $X_2 \leftarrow x_3(2x_4 - p_0x_3) + x_2(p_2x_2 + 2p_3x_1);$ 
        $X_3 \leftarrow x_4^2 + x_3(p_1x_3 + 2p_2x_2 + 2p_3x_1) + p_3x_2^2;$ 
        $X_4 \leftarrow x_4(p_0x_4 + 2p_1x_3 + 2p_2x_2 + 2p_3x_1) + x_3(p_2x_3 + 2p_3x_2)$ 
6:   else if
7:      $X_1 \leftarrow x_2(2x_3 - p_0x_2) + x_1(2x_4 - 2p_0x_3 - 2p_1x_2 - p_2x_1);$ 
        $X_2 \leftarrow x_3^2 + x_2(2x_4 - 2p_0x_3 - p_1x_2) + p_3x_1^2;$ 
        $X_3 \leftarrow x_3(2x_4 - p_0x_3) + x_2(p_2x_2 + 2p_3x_1);$ 
        $X_4 \leftarrow x_4^2 + x_3(p_1x_3 + 2p_2x_2 + 2p_3x_1) + p_3x_2^2$ 
8:   end if
9: end for
11: return  $X_1, X_2, X_3, X_4$ 

```

As is seen from the algorithm we need $6m$ multiplications and $4m$ squarings to obtain the terms $X_N, X_{N+1}, X_{N+2}, X_{N+3}$. Thus, by this algorithm and the formula (44) we can effectively compute an arbitrary fourth-order recurrence.

5. Computation of linear recurrence sequences of any order

Let $\{W_k(a_0, \dots, a_{n-1}; p_0 \dots p_{n-1})\}$ be an n th-order linear recurrence defined by the relation $f_{k+n} = \sum_{i=0}^{n-1} p_i f_{k+n-1-i}$, with the initial values $W_i = a_i$ ($0 \leq i \leq n-1$). Let $\{X_k(p_0, \dots, p_{n-1})\}$ be the sequence that is derived from $\{W_k\}$ if $a_{n-1} = 1$ and the other $a_i = 0$. Using the matrix method as in Lemma 2 and by mathematical induction we get the following formulas

$$W_k = a_{n-1}X_k + \sum_{j=1}^{n-1} \left(p_j \sum_{i=0}^{j-1} a_{n-2-i} X_{k-j+i} \right), \quad (58)$$

$$W_k = \sum_{j=0}^{n-1} \left(a_{n-1-i} - \sum_{i=0}^{n-j-2} a_{n-j-2-i} p_i \right) X_{k+j}. \quad (59)$$

If we put $n = 4$ in these formulas, then we obtain (43) and (44).

Repeating the arguments of the previous section we get the matrix formula

$$\begin{pmatrix} X_{2k+n-1} \\ \vdots \\ X_{2k+1} \\ X_{2k} \end{pmatrix} = \begin{pmatrix} X_{k+n-1} & n^{-2}W_{k+n-1} & n^{-3}W_{k+n-1} & \dots & {}^0W_{k+n-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ X_{k+1} & n^{-2}W_{k+1} & n^{-3}W_{k+1} & \dots & {}^0W_{k+1} \\ X_k & n^{-2}W_k & n^{-3}W_k & \dots & {}^0W_k \end{pmatrix} \begin{pmatrix} X_{k+n-1} \\ \vdots \\ X_{k+1} \\ X_k \end{pmatrix}. \quad (60)$$

Here, as above iW_k denotes $W_k(a_0, \dots, a_{n-1})$ with only one nonzero $a_i = 1$. Let $R = (r_{i,j})$ be the matrix from (60). It has a special form, see (34) and (52). Note that if we know two rows of the matrix R which have numbers of different parity, then we can get the other rows. For example, we assume we know a formula which relates $X_{2k+\ell}$ to X_{k+i} ($0 \leq i \leq n-1$), in other words we know the $(n-\ell)$ th row. If we replace k by $k+1$ and use $X_{k+n} = \sum_{i=0}^{n-1} p_i X_{k+n-1-i}$, then we get the formula for $X_{2k+\ell+2}$ that corresponds to the $(n-\ell-2)$ th row. Repeating this procedure we obtain all rows with numbers of the same parity as the parity of the $(n-\ell)$ th row. Thus, to get all formulas that will be used in the algorithm, we need to know formulas for X_{2k} , X_{2k+1} .

Using (58), (59) we can obtain for the elements of R :

$$r_{i,j} = \begin{cases} X_{k+n-1-(i-j)} - \sum_{l=0}^{j-2} p_l X_{k+n-2-l-(i-j)}, & \text{if } i \geq j, \\ \sum_{l=j-1}^{n-1} p_l X_{k+n-2-l-(i-j)}, & \text{if } i < j. \end{cases} \quad (61)$$

Also from the two last rows in (60) we can obtain the formulas which relate X_{2k} , X_{2k+1} to X_{k+i} ($0 \leq i \leq n-1$). Moreover, these formulas are of the same form as (55), (56).

$$X_{2k} = eX_{k+(n-1)/2}^2 + \sum_{i=0}^{\lfloor v \rfloor} X_{k+\lfloor v \rfloor - i} \left(2X_{k+\lfloor v \rfloor + 1 + i} - p_{2i+e} X_{k+\lfloor v \rfloor - i} - 2 \sum_{j=0}^{2i-1+e} p_j X_{k+\lfloor v \rfloor + i - j} \right), \quad (62)$$

$$X_{2k+1} = p_{n-1} X_k^2 + (1-e) X_{k+n/2}^2 + \sum_{i=0}^{\lceil v \rceil - 1} X_{k+\lceil v \rceil - i} \left(2X_{k+\lceil v \rceil + 2 + i} - p_{2i+1-e} X_{k+\lceil v \rceil - i} - 2 \sum_{j=0}^{2i-e} p_j X_{k+\lceil v \rceil + 1 + i - j} \right). \quad (63)$$

Here, $e = n \bmod 2$, $v = n/2 - 1$. These formulas are the generalization of (18), (19), (39), (38), (55), (56).

Now we can give the scheme of an algorithm for computing an n th-order linear recurrence $\{W_k(a_0, \dots, a_{n-1}; p_0, \dots, p_{n-1})\}$.

The scheme.

- (i) Using $X_{k+n} = \sum_{i=0}^{n-1} p_i X_{k+n-1-i}$ and repeating the replacement of k by $k+1$ in (62), (63) without removing brackets we obtain the formulas for X_{2k+i} ($0 \leq i \leq n$). They contain the same number of multiplications as (62) or (63) (depending on the parity of i). These formulas determine the rules of transition from the terms X_{k+i} ($0 \leq i \leq n-1$) to X_{2k+i} ($0 \leq i \leq n-1$) and also to X_{2k+1+i} ($0 \leq i \leq n-1$).
- (ii) By using these formulas we obtain an algorithm for computing $\{X_k\}$ that is similar to Algorithm 3.
- (iii) To get the value W_N we need to use (59) after we have computed X_{N+i} ($0 \leq i \leq n-1$) by the algorithm in (ii).
- (iv) In order to obtain W_{N+1} we use the recurrence relation to get X_{N+n} from X_{N+i} ($0 \leq i \leq n-1$) and use (59).

Remark. To compute the N th term of an n th-order linear recurrence we need $n(n+1)/2 \log_2 N$ multiplications⁴. Let us prove this. When n is even, the formulas for X_{2k+2i} ($0 \leq i \leq n/2$) contain $n/2$ multiplications⁵ and formulas for $X_{2k+2i+1}$ ($0 \leq i \leq n/2-1$) contain $n/2+1$ multiplications⁶. It is clear that each step of the algorithm needs $n/2$ formulas of the first type and $n/2$ formulas of the second type. So to compute X_{2k+i} ($0 \leq i \leq n-1$) or X_{2k+1+i} ($0 \leq i \leq n-1$) using X_{k+i} ($0 \leq i \leq n-1$) we need $n(n+1)/2$ multiplications. Thus, to compute X_{N+i} ($0 \leq i \leq n-1$) we need $n(n+1)/2 \log_2 N$ multiplications. Since (59) does not contain “big” multiplications, the above statement is proved for even n . The proof for odd n by analogous.

We have written a computer program based on the above scheme. On the next page we will give the implementation of it in *Mathematica*⁷. The function **AnyOrderRecurrence** $[a, p, N]$ returns $W_N(a_0, \dots, a_{n-1}; p_0, \dots, p_{n-1})$, where a , p are strings of length n and N is a positive integer.

⁴We use such a complexity model that multiplications involving p_i are similar to additions.

⁵Since they were derived from (62) without removing brackets.

⁶Since they were derived from (63) without removing brackets.

⁷Version Number: 10.4.0.0

```

AnyOrderRecurrence[a_, p_, N_] :=
(
  n = Length[a]; m = Length[p]; (* The lengths of the input arrays *)
  If[n == m ≥ 2 && N ≥ 0 && IntegerQ[N],
    (
      If[N ≤ (n - 1), Return[a[[N + 1]]];]; e = Mod[n, 2];

      str = IntegerDigits[N, 2]; (* The binomial expansion of N *)
      Do[X[i] = 0, {i, 1, n - 2, 1}]; X[n - 1] = 1; X[n] = p[[1]]; (* The initial values *)
      sR = Table[u[i] → u[i + 1], {i, 1, n}]; (* The string for a replacing *)
      If[n == 2, (
        (* The transition rules when the order equals 2 *)
        f[1] = u[1] (-p[[1]] u[1] + 2 u[2]); f[2] = p[[2]] u[1]^2 + u[2]^2;
        If[IntegerQ[√-p[[2]]], f[2] = (u[2] - √-p[[2]] u[1]) (u[2] + √-p[[2]] u[1])];
        f[3] = u[2] (p[[1]] u[2] + 2 p[[2]] u[1]);),
        (
          f[1] = e u[(n + 1) / 2]^2 + (* The rule for computing X2k *)
          ∑i=0⌊n/2⌋-1 u[⌊n/2⌋ - i] (2 u[⌊n/2⌋ + 1 + i] - p[[2 i + e + 1]] u[⌊n/2⌋ - i] - 2 ∑j=02 i - 1 + e p[[j + 1]] u[⌊n/2⌋ + i - j]);
          (* The rules for computing X2(k+i) 1 ≤ i ≤ ⌊n/2⌋ *)
          Do[f[2 j + 1] = ∑i=1⌊n/2⌋ Simplify[Part[f[2 j - 1] /. sR /. u[n + 1] → ∑k=0n-1 p[[k + 1]] u[n - k], 1]], {j, 1, ⌊n/2⌋}];
          f[2] = p[[n]] u[1]^2 + (1 - e) u[(n + 2) / 2]^2 + (* The rule for computing X2k+1 *)
          ∑i=0⌊n/2⌋-2 u[⌊n/2⌋ - i] (2 u[⌊n/2⌋ + 2 + i] - p[[2 i + 2 - e]] u[⌊n/2⌋ - i] - 2 ∑j=02 i - e p[[j + 1]] u[⌊n/2⌋ + 1 + i - j]);
          (* The rules for computing X2k+2i+1 1 ≤ i ≤ ⌊n/2⌋ *)
          Do[f[2 j + 2] = ∑i=1⌊n/2⌋+1 Simplify[Part[f[2 j] /. sR /. u[n + 1] → ∑k=0n-1 p[[k + 1]] u[n - k], 1]], {j, 1, ⌊n/2⌋}];
        );
      Do[Do[x[i] = X[i], {i, 1, n, 1}]; (* The temporary values xi *)
        If[str[[j]] == 1, (
          (* The transition when a new X1 compute by the formula for X2k+1 *)
          Do[X[i] = f[i + 1] /. u → x, {i, 1, n - 1, 1}]; X[n] = f[n + 1] /. u → x,
          (
            (* The transition when a new X1 compute by the formula for X2k *)
            Do[X[i] = f[i] /. u → x, {i, 1, n, 1}]), {j, 2, Length[str], 1}];
      Return[∑j=0n-1 (a[[n - j]] - ∑i=0n-j-2 a[[n - j - 1 - i]] p[[i + 1]]) X[j + 1] (* The final value WN *)],
      Print["Check the input"])]

```

Acknowledgments. The author would like to thank K.A. Sveshnikov for his interest and valuable discussions. Also, the author especially grateful to A. Bostan for pointing out Fiduccia's paper and for the evidence that our algorithm is one particular way of implementing Fiduccia's algorithm, where modular polynomial squarings are hard-coded.

References

- [1] C. M. Fiduccia, An efficient formula for linear recurrences, *SIAM Journal on computing*, **14.1** (1985), 106-112.
- [2] J. C. P. Miller, D. S. Brown, An algorithm for evaluation of remote terms in a linear recurrence sequence, *The Computer Journal*, **9.2** (1966), 188-190.
- [3] D. Gries, G. Levin, Computing Fibonacci numbers (and similarly defined functions) in log time, *Information Processing Letters*, **11.2** (1980), 68-69.
- [4] T. Koshy, *Fibonacci and Lucas numbers with applications*, Vol. **51**, John Wiley & Sons, 2011.
- [5] C. A. Reiter, *Matrix View of Horadam Numbers*, 1999.
- [6] D. Kalman and R. Mena, The Fibonacci numbers: exposed, *Mathematics Magazine*, **76.3** (2003), 167-181.
- [7] M. Joye and J.-J. Quisquater, Efficient computation of full Lucas sequences, *Electronics Letters*, **36.6** (1996), 537-538.
- [8] A. F. Horadam, Basic properties of a certain generalized sequence of numbers, *The Fibonacci Quarterly*, (1965) **3.3**, 161-176.
- [9] A. F. Horadam, Special properties of the sequence $W_n(a, b; p, q)$, *The Fibonacci Quarterly*, **5.4** (1967), 424-434.
- [10] E. L. Roettger, H. C. Williams, and R. K. Guy, Some primality tests that eluded Lucas, *Designs, Codes and Cryptography*, **77.2-3** 2015, 515-539.
- [11] D. Bleichenbacher, W. Bosma, and A. K. Lenstra, *Some Remarks on Lucas-Based Cryptosystems*, Annual International Cryptology Conference, Springer Berlin Heidelberg, 1995, 386-396.
- [12] S. Rabinowitz, Algorithmic manipulation of third-order linear recurrences, *Fibonacci Quarterly*, (1996) **34**, 447-463.

- [13] G. Gong, *An efficient algorithm for exponentiation in DH key exchange and DSA in cubic extension fields*. Faculty of Mathematics, University of Waterloo, 2002.